
Ça marche toujours pas, vous vous foutez de moi !? On perd des milliers de visiteurs là !
Pss... Le client a toujours le problème, c'est chaud là...
Impossible, c'est corrigé depuis 1h, j'ai tout vérif...
Hum...
Oups ! Si ! C'est de ma faute, c'était pas bien passé en prod !
Grrr...
Mais attends, c'est bon là, j'viens d'réenvoyer
Bah qu'est-ce que je dis au client moi maintenant !?
Ben... Comme d'hab...
Oui, allô Monsieur ? Ça bug toujours ?
Ah voilà, enfin ! Ça marche ! C'était quoi le bug ?
On a rien touché, à priori, c'était un simple problème de cache de votre côté...
Ah, tant mieux. Merci bien
CommitStrip.com

TCM://
THE CODING MACHINE

You will no longer say "clear the cache"
in a small hint of shame.

TheCodingMachine

Cache pattern

Point du vendredi – 16 sept. 2022

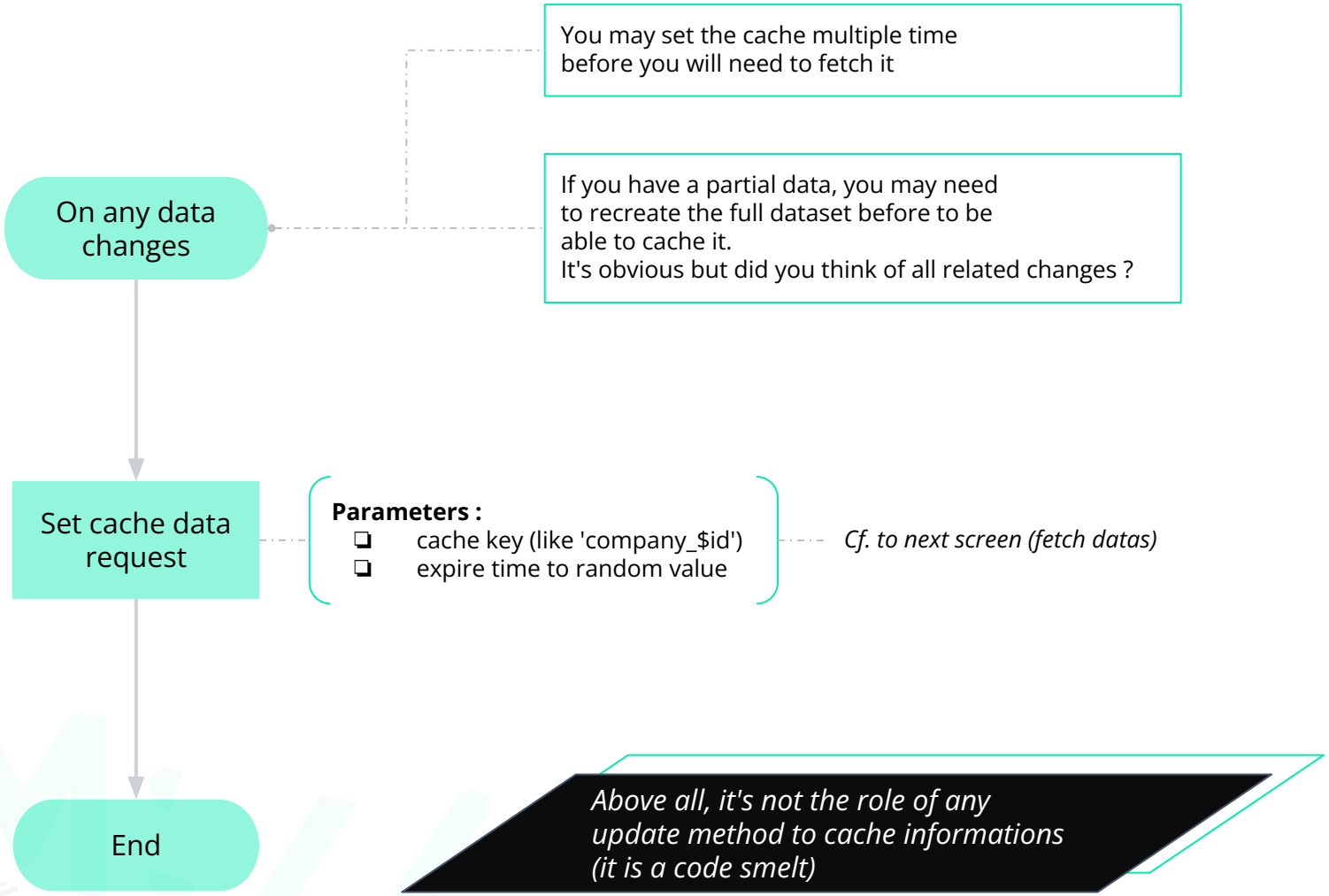
MIO@TheCodingMachine

Table of contents

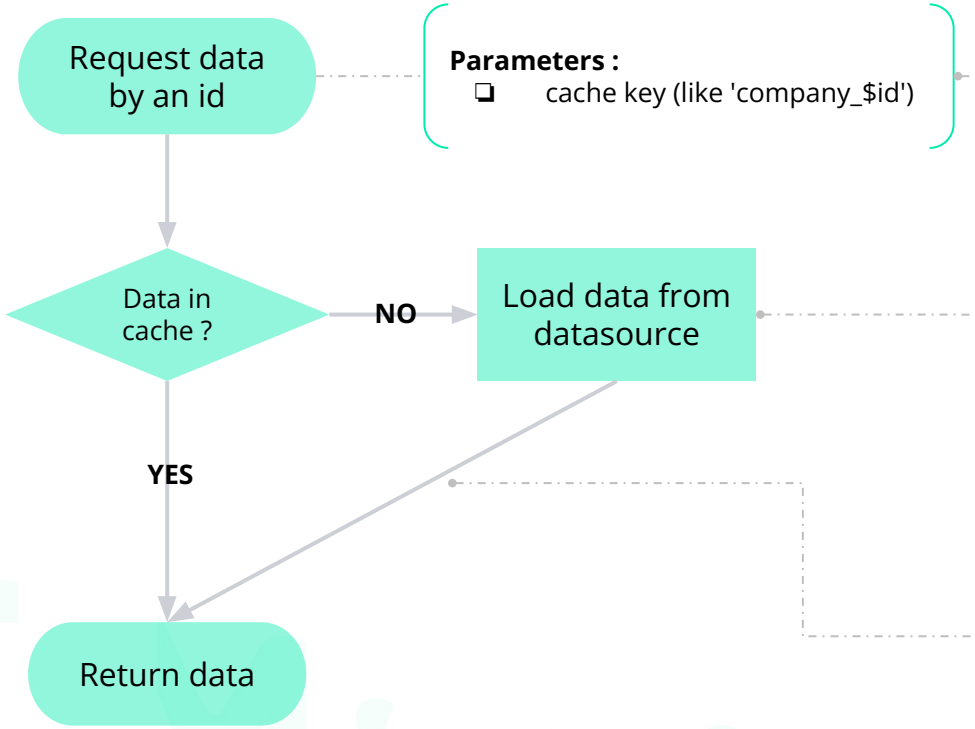
01

WRONG WAY

Store cache


You may set the cache multiple time
before you will need to fetch it
If you have a partial data, you may need
to recreate the full dataset before to be
able to cache it.
It's obvious but did you think of all related changes ?
On any data changes
Set cache data request
Parameters :
❏ cache key (like 'company_$id')
❏ expire time to random value
Cf. to next screen (fetch datas)
End
Above all, it's not the role of any
update method to cache informations
(it is a code smelt)

Fetch



Parameters :

- ❏ cache key (like 'company_$id')

- ❏ In any update of structure of data, you will need a full cache clear and application will need to rebuild all.

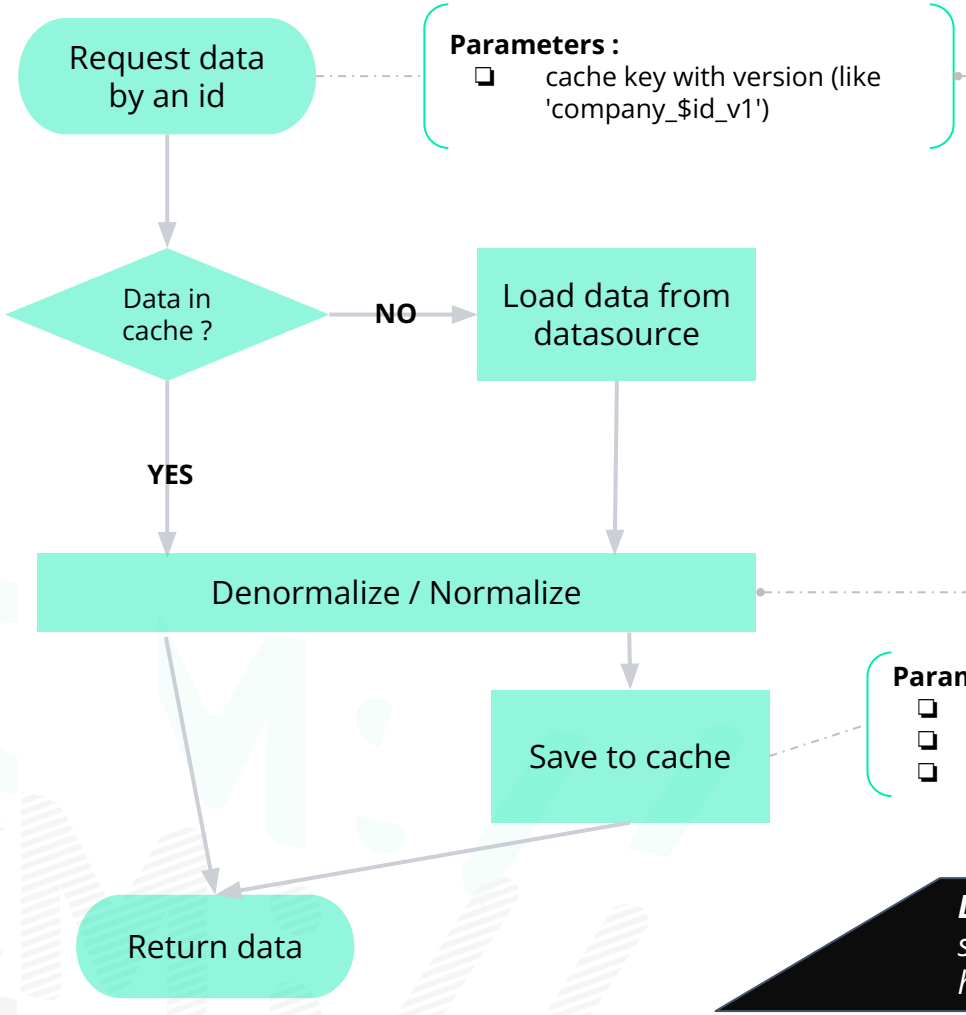
- ❏ You will have outdated value the most of the time (because of random expire time, but it's not related to an action)
- ❏ You can't trust your cached data so you will place some parameters to allow to fetch datas without cache (it's bad because if you don't use cache for some part of application, you lose all advantages of it).

- ❏ Here you may store data to the cache but you may need for that to duplicate code for cache storage (and the most of time, you forget this part)

02

GOOD WAY

Fetch cached



Parameters :
- cache key with version (like 'company_$id_v1')

- In any update of structure of data, you don't need to invalidate cache, next request will automatically request for 'company_$id_v2'

- Optional part useful if your cached information contain recursives datas, or data who can be recomputed at each request (on your concrete class of entity, you can implement serialisable)

- With "tags" you can link your cached data to many entities. It's useful for invalidation.
  So tags a lot : one tag for company, one for each sub companies, etc.
- Expire time will be only used to clear unused information from cache storage

Parameters :
- cache key with version
- cache tags list
- very long expire time (30 days)

Do not cache too much ! *A request to cache storage take time ! With too much you may have a* ***slower*** *application.*

TCM://

Invalidate

On any data changes

Find all related entities

For each, just invalidate the **TAG**

- ❏ You can implement it as event on your ORM for all entities (you don't need to know if there is cache feature for them)
- ❏ Do not forget DELETE
- ❏ You may need also to invalidate for CREATE (for systems where id is provided by another systems : email per example). It's a security.

- ❏ For each entities generate all related tag you may use on your application.
- ❏ A tag factory will help you to centralise all onto complex systems.

- ❏ Do not invalidate key, invalidate only TAG. With this way, you will invalidate all cached data related to an entity. Per exemple a company with some related subcompagnies, if a subcompany change, the root company related cache may change too.

If you need to be very comfortable, you can execute it on a transaction mode (if invalidation fail, rollback changes)

Cache prefetching (optional)

By any job feature

- ❏ You may use cronjob, consumer, event consumer, manual call, command line during deployment...
- ❏ Just avoid to do that during the real processes : this task have to be executed in background mode.
- ❏ If any call to the application request a data who are not yet prefetch, it's not a trouble (there will be cached by this request).

Fetch maximum amount of data in one call

- ❏ Any call to an external source (like sql server) require some time to execute. If you make one call for multiple datas, you avoid that.
- ❏ But... paginate results : you have to balance between speed and memory limit of your process.

For each, apply cache storage

- ❏ To apply cache storage, you may same method as you use for the regular way (you just need to split you getter in two parts).

Prefetching is not a standard requirement. You may use it to speed up the first load... but your server will load lot of unrequired data (use that only for specifics cases or if fetch lot of information in same request is very faster than request one by one).

03

SOME TIPS

Security implications

- Becareful about “context” : some informations are generated for only some publics. In this case, add a suffix in the key (about something related the right/role) or do not cache confidentials informations (remove/add them during normalisation/denormalisation).
- In local cache (like browser cache), you may store some confidentials information… but a local computer can be lost or stolen…
- A cache server may have also security breach. So if you expose some very confidentials informations, please double check security of cache storage (redis, memcache, file, etc). Lot of security breach are linked to memory overflow or request forgery in cache server.

It’s very difficult to be in security with caching feature… but it’s your job !

What datas to cache

- Save request who need lot of time to return (per example from an external service or very complex SQL request).
- Request called very often (something you need everywhere, something you need to share for lot of user in same time…)
- To make an information immutable for a while : per example an exchange rate (to have the same rate for one hour).

Where to store the cache

- Filesystem or memory : but in these case, you cache is not stateless.
- Redis, memcache, etc : the most used way (very easy to manage)
- In your SQL database : it's named denormalization because the a "good" should not have any information who can be calculate from others tables… but perfectionism can be counter-productive (in a e-commerce application, it's easier to have all order total in database than recalculate each amount row by row).
- Semaphores : it's allow to share the same memory pointer for multiple processes. It's used the most of the time for lock feature and it's not stateless (but very efficient if your information change very fast and are used by multiple processes in same time).
- In static variable in php : it's not very useful but it's very easy to implement. But each request will drop them (it's just a global variable).

Cache and dev env

In dev environment :

- In dev environment, you can have a variable to disable cache for a **temporary** debug. But you should use cache even during development.
- Clear the cache after each switch branches / reload fixtures.
- For unit test, do not forget to clear them in setUp.

In staging environment :

- Staging is just another production environment with fake data.
- You should never clear the cache : if you need it, you made a mistake during implementation and the production will have the same trouble.

How to manage remote information

When an information is provided by an external source, you may have difficulties to invalidate the cache. Some possibilities :

- ❏ Manage a webhook : an API call from external source to your application to invalidate
- ❏ Create a key based on remote information easiest to fetch. Per example, make a call to have a checksum of informations and make the call(s) to have the full information only if this checksum is not stored on your cached data.
- ❏ Manage a cron task who will fetch all changes from a last call an invalidate information for each changes (but with this method, you cached data can be outdated from the last call to the endpoint to fetch changes : so please inform user in UI about date of last update).
- ❏ Then… if you can't do anything to have an invalidation managed by the remote system… you can't do anything valid for caching. If it's not too sensible, just take in account that your information IS outdated (if it may, it is).

Docs

- https://symfony.com/doc/current/components/cache.html#basic-usage-psr-6
- https://laravel.com/docs/8.x/cache
- https://developer.wordpress.org/reference/classes/wp_object_cache/ (not very usefull : please customize it)

Merci !
Xie xie !
Obrigado !

Merci à Pauline qui avait préparé le template pour cette présentation il y a quelques mois !

Template

step 14pt

optional 10pt

start 12pt

- text 1 ... 10pt
- text 2 ...
- text 3 ...

Comment ... 10pt

step 12pt

Parameters :
- ... 10pt
- ...

Cf. ... 10pt

Condition 10pt

Condition 10pt

end 12pt

Description... 12pt

other step 12pt

other step 12pt